Practical Application of Verified Boot

Verified Boot and it’s common pitfalls

Rouven Czerwinski – r.czerwinski@pengutronix.de

About me

Rouven Czerwinski

Pengutronix e.K.

Emantor

rcz@pengutronix.de

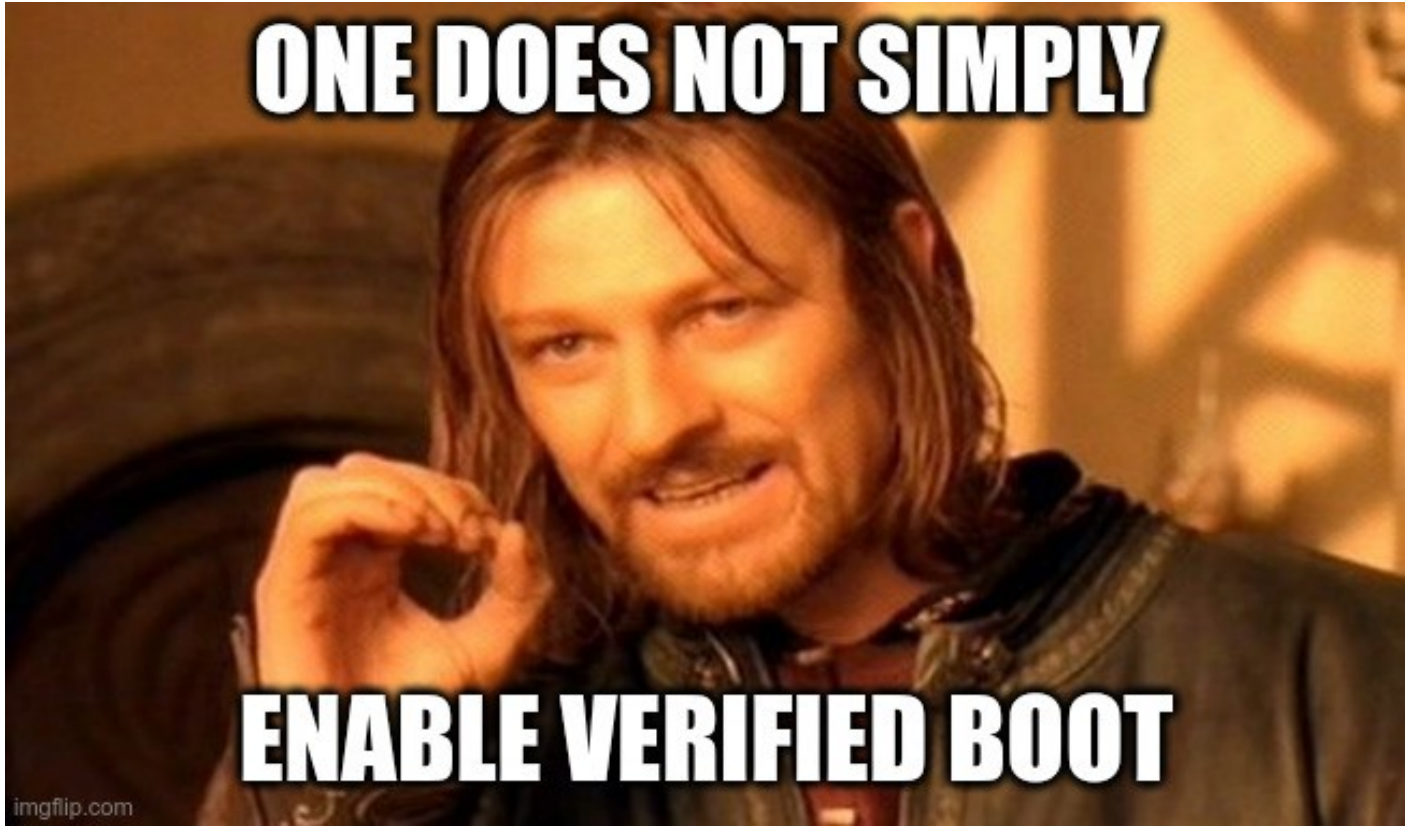
Testing

OP-TEE & Security

System Integration

Linux Media

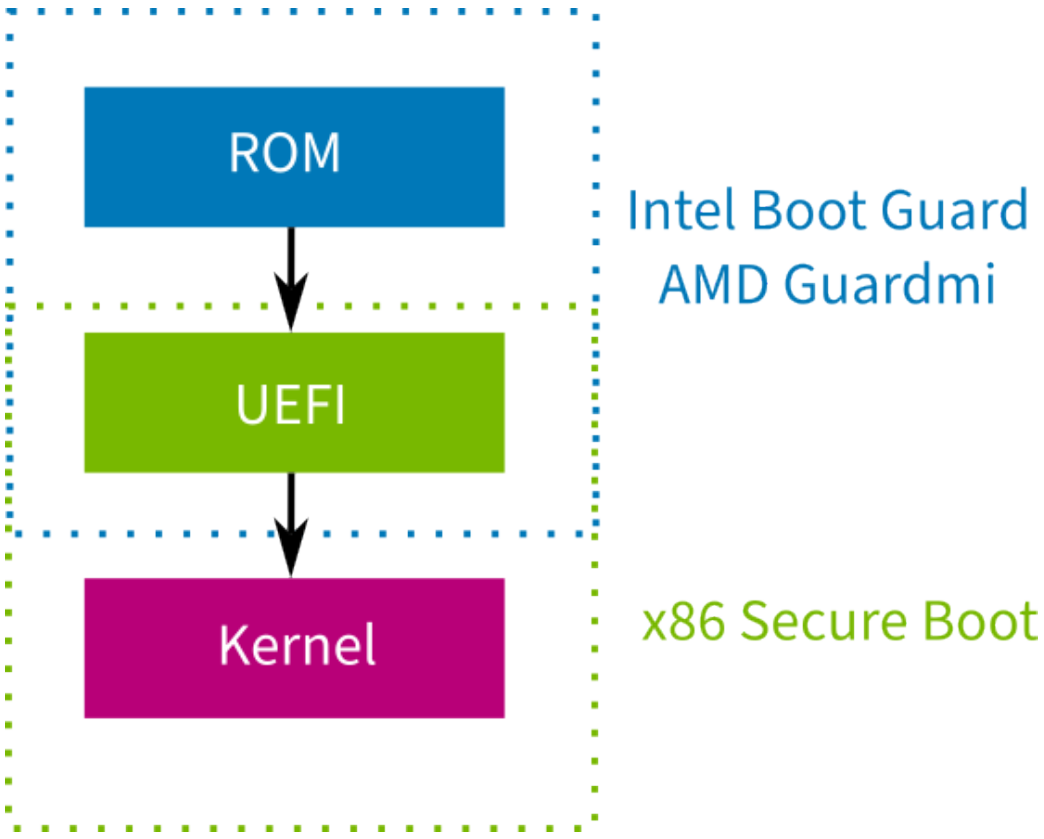
Boromir says it isn’t easy



X86 & ARM

- X86: Secure Boot
- ARM: Vendor Implementation
  - NXP: High Assurance Boot (HAB), Advanced High Assurance Boot (AHAB)
  - Rockchip: NDA-requirement
  - TI: called Secure Boot, has nothing to do with X86 implementation

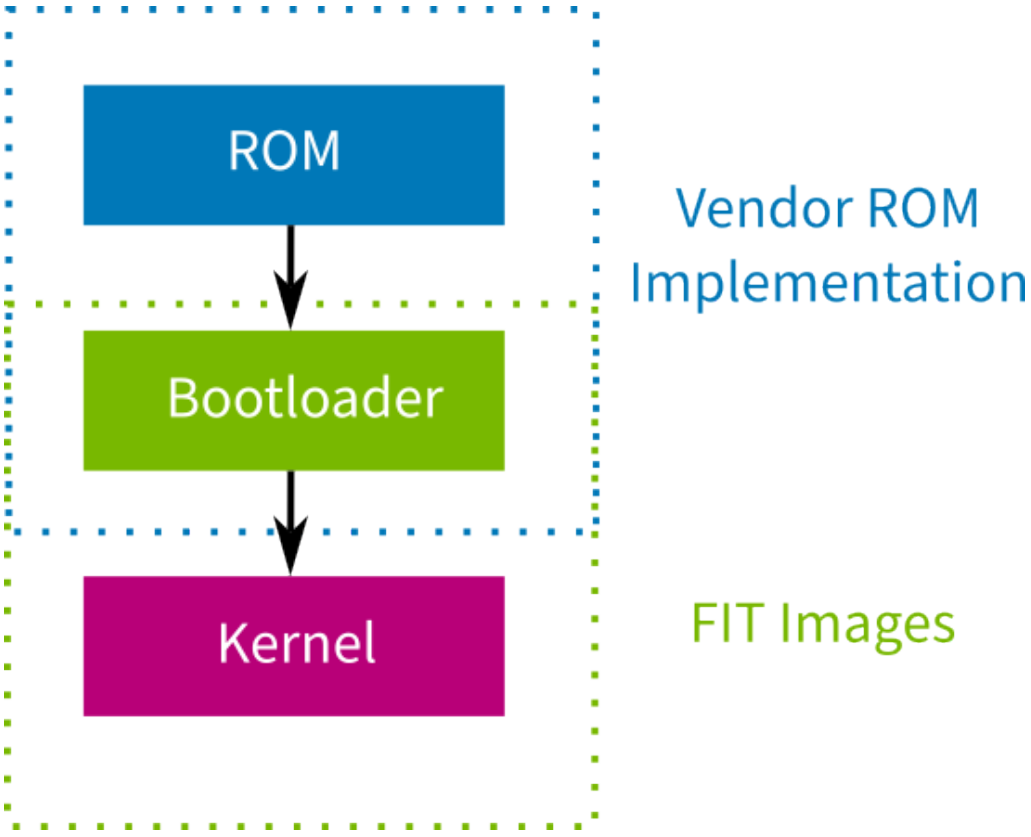
X86 Secure Boot



x86 Secure Boot?

- X86 Secure Boot works with it own CA
  - Tool for CA Implementation: sbctl
  - You can deploy your own certificates
- License problems & complexity stay the same
- UEFI Password

ARM Secure Boot



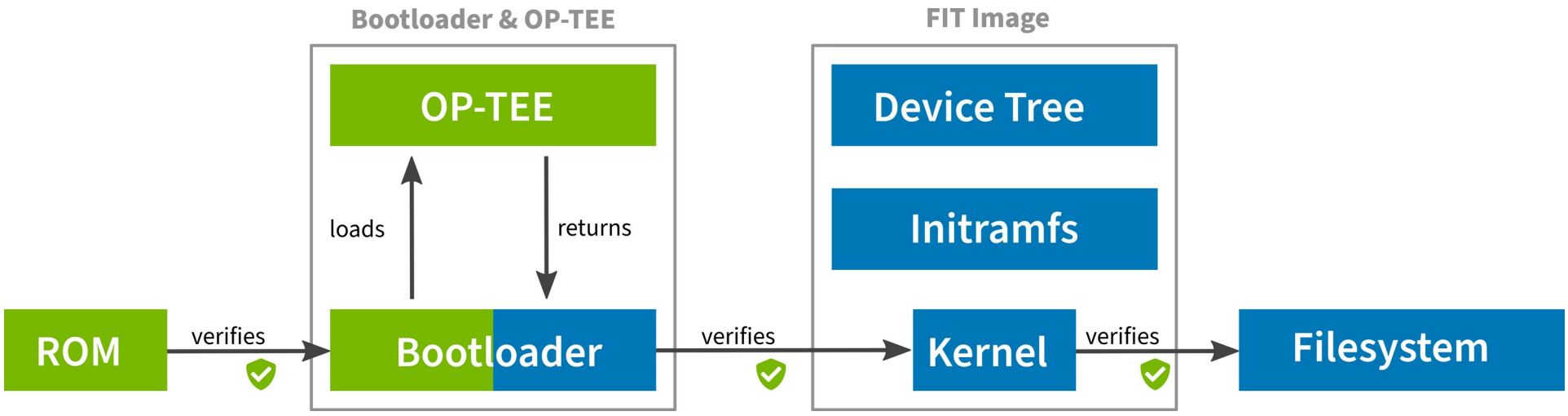
ARM Verified Boot

- Almost all implementations require fuse burning
- Fuse: one-time programmable to one
- i.e. NXP: burn hash of the CA certificates
- → ONE-TIME operation
- → NO new deployment of certificates possible (as x86 secure boot provides)

Project 2019-2021: once upon a time

- ARM Embedded i.MX6UL Plattform
- High Assurance Boot (HAB)
- OP-TEE to save authentication information for the cloud
- AppArmor
- RAUC Updates
- Yocto
- Nitrokey HSM für Verified Boot Keys

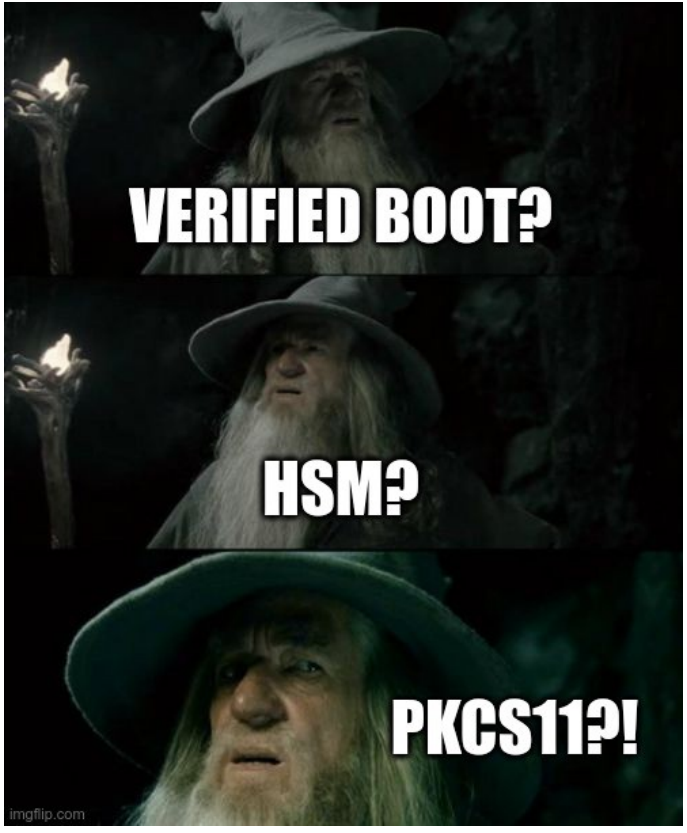
Booting: Example for ARM systems



2022: Customer team no longer exists

- Team members left the company
- Company site was sold off
- Yocto was left in the state of 2021
- Questions to customer for further support/updates went unanswered

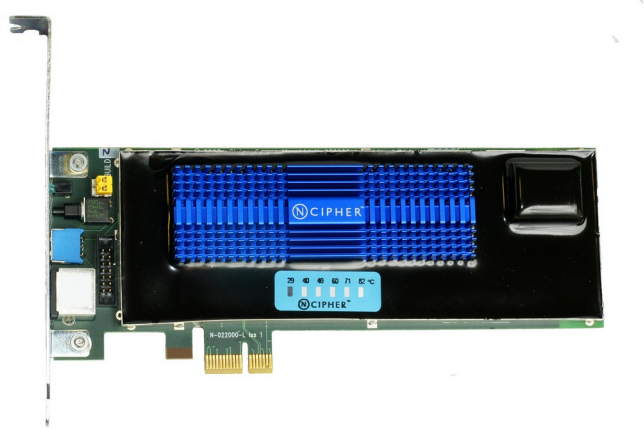
End of 2023: New Team



PKCS#11

- Standard Programming Interface
  - Hardware Security Modules (Yubikey, Nitrokey,...)
- i.e. generate assymmetric key pairs
- Private part is marked as non-exportable
- Supported in:
  - Chromium, nginx, SSH, wpa_supplicant, curl, evolution, Linux module signing, FIT images, RAUC, code signing
  - Usage often in conjuction with OpenSSL/GnuTLS

PKCS#11 HSMs



CC BY 3.0 Alexander Klink (Wikimedia Commons)

Ncipher PCIE Card

Yubikey (Example)

Nshield Connect 45 (19“ Hardware)

End of 2023: everything has to be different

- New Team
- USB HSM to be replaced with Cloud HSM
- Software Bill of Materials (SBOM) requirement
- → Yocto-Update required
- „Excuse me, but do you know how this works?“

Conclusion: project knowledge is essential!

- For normal devices this can be clawed back
- For verified-boot devices not even bring-up works
  - What kind of board do I have? Production? Which keys were used?
  - How do I update the system?
  - Bootloader is locked, no reconfiguration possible
- Absolute bare minimum: hand-over workshop
- Better: hand-over period where questions can be answered by the old team

„The contractor does it.“

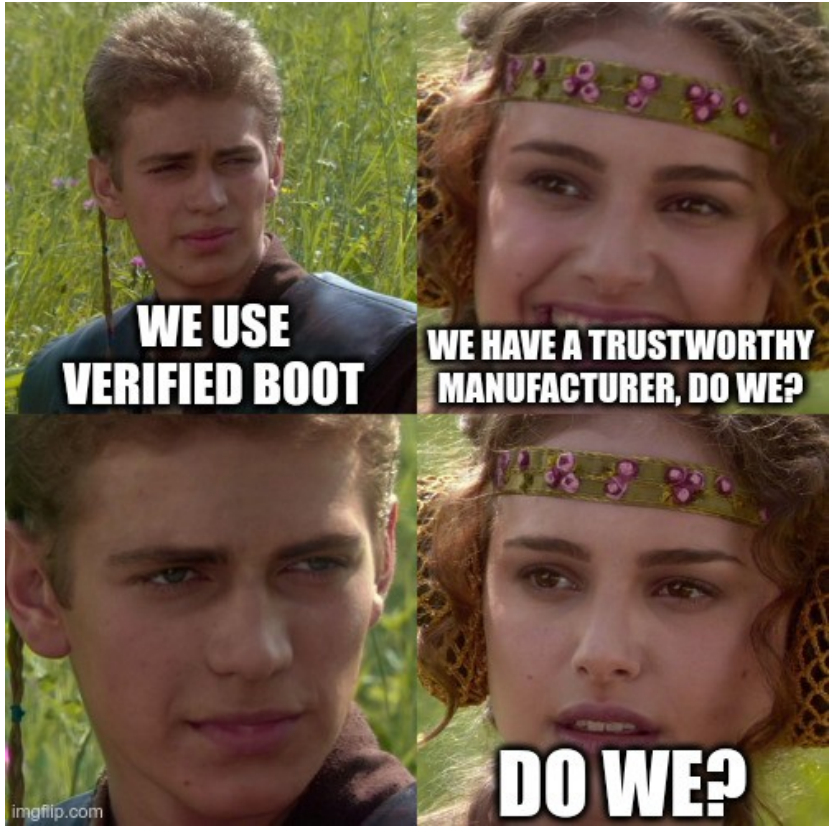
- Verified Boot can NOT be done by a contractor alone
- Technical decisions require project knowledge, sometimes outside the contractors domain of expertise
- Contractor can’t make these decisions
- If the contractor is no longer involved, decisions HAVE to be done by the device vendor (YOU!)
- Key management also needs to be done by the device vendor

Development on device

- „Open“ reconfigurable bootloader which can boot anything
- Two Choices:
  - Development Bootloader signed with release keys
    - Advantage: released device == development device
    - Disadvantage: Bootloader unlocks EVERYTHING
  - Development bootloader with development keys
    - Advantage: Bootloader buildable in the CI
    - Disadvantage: special development devices

Oops, I put the dev configuration into CI

Device provisioning



Example: provisioning

- Verified Boot is done, implemention tested, ready for release
- Steps before deploying software on device:
  - Enable verified boot
  - Initialize OP-TEE (eMMC RPMB), setup pSLC
  - Provision device private key
  - In a trusted environment!

Key division

- How many keys do we generate?
  - One key for all devices?
  - For each revision?
  - For each production run?
- Document why!

Documentation

Documentation

- Document:
  - How release keys are provisioned
  - How the keys were created
  - How development devices can be distinguished from release devices
  - How to handle field return devices
  - …
  - Test and verify your instructions periodically!

Licensing Problems

- GPLv3 contains „TIVO-Clause“
- As someone who bought a device I am entitled to all information to run modified GPLv3 software on the device
- → Entitlement to secure boot keys
- Solution:
  - No GPLv3 Software → Bad, harder and harder to avoid GPLv3 software
  - Unlock-Mode: unlock bootloader, destroy sensitive key material

Handling devices returned from field

- Unique handling for each Verified Boot implementation
- i.e. NXP: special bootloader with SoC unique ID
- → HAB/Verified Boot Disabled
- Complex process, may need automation for support

Yocto: signing.bbclass

- Author: Jan Lübbe
- Usage of PKCS#11 within recipes (HSM access)
- Upstream since mickledore (contained in scarthgap!)
- Usage examples:
  - Bootloader
  - Update-Signing
  - FIT-Image Signing
- Integration example: Github Link

Yocto: signing.bbclass

```
#TOKEN_ARG = "serial=DENK0200554"

# Select HSM by url-encoded label (for example "UserPIN (Dev)")

#TOKEN_ARG = "token=UserPIN%20%28Dev%29"


# Setup the token pin by replacing "123456"

#TOKEN_PIN = "pin-value=123456"


# PKCS11 module to be used, opensc for NitroKey HSM

#SIGNING_PKCS11_MODULE = "/usr/lib/x86_64-linux-gnu/opensc-pkcs11.so"
```

Signing.bbclass

```
SIGNING_PKCS11_URI[imx_habv4_srk1] = "pkcs11:${TOKEN_ARG};object=ptx-dev-SRK1&${TOKEN_PIN}"
SIGNING_PKCS11_URI[imx_habv4_srk2] = "pkcs11:${TOKEN_ARG};object=ptx-dev-SRK2&${TOKEN_PIN}"
SIGNING_PKCS11_URI[imx_habv4_srk3] = "pkcs11:${TOKEN_ARG};object=ptx-dev-SRK3&${TOKEN_PIN}"
SIGNING_PKCS11_URI[imx_habv4_srk4] = "pkcs11:${TOKEN_ARG};object=ptx-dev-SRK4&${TOKEN_PIN}"

#SIGNING_PKCS11_URI[imx_habv4_csf1] = "pkcs11:${TOKEN_ARG};object=ptx-dev-CSF1_1&${TOKEN_PIN}"
#SIGNING_PKCS11_URI[imx_habv4_img1] = "pkcs11:${TOKEN_ARG};object=ptx-dev-IMG1_1&${TOKEN_PIN}"

SIGNING_PKCS11_URI[fit] = "pkcs11:${TOKEN_ARG};object=ptx-dev-fit&${TOKEN_PIN}"

SIGNING_PKCS11_URI[rauc] = "pkcs11:${TOKEN_ARG};object=ptx-dev-rauc&${TOKEN_PIN}"
```

Conclusion

- Documentation, Documentation, Documentation
- Hand-over-process when the contractor leaves or a new team member needs on-boarding
- Be careful abort:
  - Provisioning
  - Field returned devices

Don’t forget the application!

But why don’t we use TPM?